BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/604,987	06/28/2000	Srivatsan Parthasarathy	MS146910.1	6447

27195 7590 07/15/2004

AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 07/15/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/604,987	PARTHASARATHY ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 April 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 4/15/2004.

As indicated in Applicant's response, claims 6, 15 have been amended. Claims 1-35 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-8, 18, and 23-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051 (hereinafter Renaud), in view of Buxton, USPN: 6,182,279 (hereinafter Buxton).

As per claim 1, Renaud discloses a method for integrity checking employable by application programs at runtime (e.g. Fig. 6; col. 9, line 40 to col. 10, line 37), such method comprising:

providing an assembly (e.g. *data structure 300* – Fig. 3a) that contains a list of versioned modules (e.g. Fig. 3a-b; *data files 304-314, version of the file* - col. 6, lines 46-64; col. 7, lines 31-36 – Note: Java class files or applets are equivalent to modules) that make up the assembly;

providing a manifest (e.g. *signature file 302* – Fig. 3b) with a hash value of data related to at least one module of the list of modules (e.g. *identifier, one-way hash* - col. 7, lines 15-27).

But Renaud does not specify that the hash of the list of modules making up assembly is a hash of the contents of at least one module. Renaud, however, teaches arranging modules or file data in such way that digital signature is combined with digest encryption algorithm and that the manifest can include data related to the modules comprising the assembly (e.g. *data relating to each data file, MD5, MD2* - col. 7, lines 24-54) to facilite processing and security checking. Buxton, in a system to load objects in an object-oriented user interface environment like Renaud's activating of applets and Java files via an GUI screen (Renaud: Fig. 5a), discloses template as stage for storing loaded control components for a container application (e.g. *DLL storage 205, OLE Libraries, Template 208* - Fig. 2 – Note: Loading dynamic libraries and linking control objects to OLE/COM applications is equivalent to runtime linking); such templates, i.e. manifest, (*template* – Fig. 2) for associating components hashed contents for tracking of object identification (e.g. col. 12, lines 1-15; cols. 19-20); and integrity checking (e.g. Fig. 7-8B) using the hash of the contents of the components to be loaded, i.e. control components loaded by a *component loader* via a template execution with each component being a self-contained module (e.g. Fig. 2; Fig. 3A; col. 9, lines 8-25; col. 8, lines 49-63). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the integrity checking of versioned modules listed in the manifest by Renaud such that such manifest lists not only the data modules hashed identifiers or data related to those modules but also a hash of the contents of such modules just as taught by Buxton. Renaud's hashing of module identifier differs from Buxton's module content hashing in that the former requires less resources, but in case processing resources are available so to enable encryption of the whole module content as

in Buxton's system, one ordinary skill in the art would be motivated to do so is that submitting the modules as a whole via a digest encryption upon the time when the modules has been originally created would make it more accurate for subsequent integration checking as intended by the file download and identifier checking by Renaud.

As per claim 2, in view of the combined teachings of Renaud and Buxton from claim 1, the limitation of providing a hash of the contents each module that constitutes the assembly would have been obvious because the integrity checking would fulfill the same purpose as matching the integrity of such module at runtime, the difference being matching hashed contents would require more resources but would yield integrity of data on the data itself as mentioned in claim 1 above.

As per claim 3 and 4, Renaud further discloses providing identity information in the manifest (e.g. Fig. 3b, *additional data* – col. 6, lines 55-64) as well as publisher and version information.

As per claim 5, in reference to claim 1, Renaud discloses hash of the identifiers of versioned modules in the assembly and suggests placing a signature (or hashed representation)of the manifest itself on top of the manifest with all other module identifiers at the end of the manifest (e.g. Fig. 3B), but **fails to disclose** providing a hash of the contents of such assembly at the end of the assembly. But this limitation to provide hash of the contents of modules has been addressed in claim 1 above. Hence it would have been obvious for one of ordinary skill in the art at the time the invention was made to add such hash of contents as taught by Buxton at the end of the assembly to enhance the lay out of the manifest in order to expedite information and facilitate

efficiently the integrity checking as intended per module when resources at built time are available as mentioned in claim 1 above.

As per claim 6, Renaud in combination with Buxton, discloses version number included in the manifest (Fig. 3b) and security checking of modules and comparing of signatures identifier from the manifest and generated from the module (Fig. 4, 6); but does expressly mention determining if the contents of the assembly has been modified by comparing the actual hash from the module contents with the hash stored in the manifest. Buxton, similar to Renaud, discloses if data module has not been tampered with when authenticating it against the signature of the publisher; further, Buxton teaches integrity checking to see if data is corrupted using licensing and registration mechanisms (e.g. *check license and integrity, component has been damaged* - col. 17, lines 16-44); hence suggests whether the loaded module has been the same as originally created and registered. Buxton's way of comparing to determine if the module contents as hashed have been modified is therefore implicitly disclosed. In case Renaud's hashed signature comparing does not already include a verification as to whether a module has been modified, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a integrity checking such as taught by Buxton to the data authentication by Renaud's using the version number in the manifest as mentioned above to determine if the up/down-loaded module for activation are integral with respect to their predetermined version, thus enhancing further the integrity checking of data and security control as intended by both Renaud and Buxton.

As per claims 7 and 8, Renaud discloses version and publisher information (e.g. col. 6, lines 55-64) and thereby suggests which version determination but fails to

expressly disclose if the assembly has been modified using such information (re claim 7) but this limitation has been addressed in claim 6 above. Further, Renaud does disclose whether the publisher of the assembly is (re claim 7) trustworthy (e.g. step 506 – Fig. 5, 5a; col. 9, lines 8-21) via authenticating publisher name information in the manifest(re claim 8).

As per claim 18, Renaud discloses a computer readable medium (col. 15, lines 13-30) with an executable for a runtime application program, such medium comprising

an assembly (e.g. *data structure 300* – Fig. 3a) including manifest containing a list of modules making up the assembly (e.g. Fig. 3A); and

a hash of at least one list of modules in the assembly (e.g. identifier *316, 318* - Fig. 3B; col. 7, lines 15-27).

But Renaud does not specify that a hash of the list of module in the manifest is a hash of the contents of at least one module. But this limitation has been addressed above in claim 1 using Buxton's teachings.

As per claim 23, this is a system claim version of claim 1 above; hence is rejected herein using the corresponding rejection set forth therein.

As per claim 24, this claim recites comparing hash of a module in the manifest and hash of actual module is analogous to the comparison limitation as recited in claim 6 above. Hence, the rejection in claim 6 herein applies.

As per claim 25, Renaud discloses identity, publisher and version information and Buxton discloses integrity checking of code assemblies or modules as mentioned in claims 3-4 and 6-8 when addressing trustworthiness of hash value, publisher and version information above. Based on such teachings and rationale of rejection as set forth

therein, the limitation of using identity and version information to determine if the assembly should be executed would also have been obvious using the rationale of mainly claims 6-8.

4. Claims 9, 19-21, 26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051, and Buxton, USPN: 6,182,279, as applied to claim 1, 18, 23, and further in view of Evans et al., USPN: 5,805,899.

As per claim 9, in reference to claim 1, Renaud discloses a signature or hashed representation of the manifest file (Fig. 3B) and identification of multiple sites signature (hashed representation) for a given loaded files (col. 9, lines 30-39; *Add site* - Fig. 5a) to verify the authenticity of the module in the loaded assembly (e.g. col. 4, lines 9-34) of another versioned object (Fig. 18) **but does not disclose** a manifest providing hash of another manifest upon which it depends on. Buxton teaches signature of components (re claim 1) and pointer information to control dependency of components in the aggregate of OLE components to combine at runtime template (e.g. col. 9, lines 45-67). Evans, in a method to link runtime versioned objects similar to the combination of components of Buxton's template using hash value of object identifier (Evans: Fig. 11) analogous to that such as taught by Renaud (Renaud: Fig. 4), discloses pointer information to refer to the assembly manifest of another versioned object that the assembly depends on (e.g. Fig. 16). In view of the pointer reference teachings by both Buxton and Evans, combined with the hash representation of the manifest by Renaud, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the manifest file or signature file of Renaud, with pointer information to locate the site of another manifest site or assembly that the pointing assembly depends on, such location

represented by a hashed identifier as suggested by Renaud. One of ordinary skill in the art would be motivated to do so because this would extend the security/version checking adopted by Renaud in that it reduces time to locate a referred to site of another assembly of modules, such reduction of time being enhanced by using pointing means as suggested by Buxton and furthered by Evans, to locate and verify the hashed representation of the depended-upon assembly, or hash of the manifest of the assembly pointed to by the dependent assembly.

As per claim 19, this claim includes a manifest with a list of at least one referenced assembly and a hash of manifest of such referenced assembly.

Renaud teaches representing a manifest with a hashed representation (*signature 322* - Fig. 3B) and suggests dependency on multiple location (hashed) identifiers by one assembly at verification and runtime (e.g. col. 9, lines 30-39; *Add site* - Fig. 5a; col. 4, lines 9-34; Fig. 18). In a method to bind objects at runtime using hash representation of versioned assemblies analogous to the Renaud's signature file listing, Evans discloses pointer means to refer to other manifest of other assemblies that the pointing assembly depends on (re claim 9). Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the representation of manifest by Renaud with the implementation suggested by Evans to point to other assemblies referenced to by the pointing assembly in providing a manifest with the list of referenced assemblies (or files or objects) and using therein pointer information to point to the hash representation of the manifest of the assembly referred to by such pointer because of the same benefits as set forth in claim 9 above.

As per claim 20, refer to rejection of claim 4.

As per claim 21, Renaud mentions about application classes used in a GUI window environment (e.g. Fig. 5a) to provide interactive verification of applets and Buxton discloses template storage DLL to enhance the component builder (col. 7, lines 48-61). Official notice is taken that the use of Dynamic Linked Library to effect windows application and user interface functionality was a well-known concept at the time of the invention. In case the components to build by Buxton or the assembly of code or modules by Renaud do not already include a Dynamic Linked Library (DLL), it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a DLL as one type of assembly of code in the list of assemblies as taught by Renaud/Buxton, because of the known benefits ascribed to using DLL such as portability, storage benefits and ease to use without the need for recompilation.

As per claim 26, Renaud does not expressly teach a binding policy but discloses establishing and verification of signature with version information included and access permissions protocol (e.g. Fig. 5, 5a, 7) to enable trusted communication to bind usage of received files to the secure and trusted protocols; and also teaches determining which applet is to be trusted for execution (Fig. 6). In building components using secure object verification means, Buxton further enhances Renaud's version recording by disclosing integrity checking to verify if data are not corrupted or modified (*check license and integrity, component has been damaged* - col. 17, lines 16-44). And Evans discloses binding assemblies to another version representation of another assembly and ensuring that the correct version of object gets executed (Evans: Fig. 11, 15). It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify Renaud's (enhanced by Buxton) method for integrity checking using trusted protocol of

versioned assemblies such as to enhance it with the version dependency linking and correct version determining policy by Evans and effect it such to determine which version is to be executed when another version is resident on the executing environment because this would enable alleviating fatal system errors from having uncontrolled versions dynamically competing in a same system.

5. Claims 10-13, and 27-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051, in view of Evans et al., USPN: 5,805,899 (hereinafter Evans).

As per claim 10, Renaud discloses a method for facilitating integrity checking employable by application programs at runtime (e.g. Fig. 6; col. 9, line 40 to col. 10, line 37), such method comprising:

providing an assembly (e.g. *data structure 300* – Fig. 3a) with manifest (Fig. 3b) that contains a list of assemblies (e.g. *data files 304-314, version of the file* - col. 6, lines 46-64; col. 7, lines 31-36 – Note: data files, digital stream or class files are assemblies of other sub-assemblies like subclasses or subdata) that make up the assembly;

providing a manifest (*signature file 302* – Fig. 3b)with a hash of the contents of at least one assemblies (e.g. *identifier, one-way hash* - col. 7, lines 15-27).

But Renaud does not disclose that the manifest contains a list of referenced assemblies that the assembly depends on; nor does Renaud disclose a manifest with a hash of a manifest of one referenced assembly of the list of referenced assemblies. However, Renaud teaches representing a manifest with a hashed representation (*signature 322* - Fig. 3B) and suggests dependency on multiple location (hashed) identifiers by one assembly at verification and runtime (e.g. col. 9, lines 30-39; *Add site* -

Fig. 5a; col. 4, lines 9-34; Fig. 18). In a method to bind objects at runtime using hash representation of versioned assemblies analogous to the Renaud's signature file listing, Evans discloses pointer means to refer to other manifest of other assemblies that the pointing assembly depends on (re claim 9). Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the representation of manifest by Renaud with the implementation suggested by Evans to point to other assemblies referenced to by the pointing assembly in providing a manifest with the list of referenced assemblies (or files or objects) and using therein pointer information to point to the hash representation of the manifest of the assembly referred to by such pointer because of the same benefits as set forth in claim 9 above.

As for claim 11, the limitation would also have been obvious in view of the rationale as set forth in claim 10; because providing in a manifest a hash of each of the referenced assemblies would enable more time-efficient locating of depended-upon assemblies as well as verifying their integrity and authentication as intended by Renaud.

As per claims 12 and 13, Renaud also teaches identity information in the manifest (re claim 3) and publisher and version information (re claim 4).

As per claim 27, Renaud discloses a method for facilitating integrity of assemblies employable by application program at runtime, such method comprising components for:

a first component to provide a manifest of assembly (e.g. *signature file 302* – Fig. 3b; *data structure 300* – Fig. 3a - Note : data structure 300 is assembly and signature file is manifest) that contains at least one a sub-assemblies (e.g. *data files 304-314, version of the file* - col. 6, lines 46-64; col. 7, lines 31-36 – Note: files, digital stream or class files

are sub-assemblies making up assembly structure 300) that make up the assembly, such sub-assemblies comprise a manifest (e.g. Fig. a-b – Note: file representation by identifier 316, 318 is equivalent to manifest of each sub-assemblies); and

a second component to provide the manifest with a hash of the sub-assemblies (e.g. *identifier, one-way hash* - col. 7, lines 15-27 – Note: hash representation of file identifier is hash of the manifest of the sub-assemblies).

But Renaud does not discloses that the sub-assembly in the manifest is a referenced assembly; but this limitation has been addressed using the teachings by Evans to link assemblies to other assemblies reference using the rationale as set forth in claim 10 above.

Nor does Renaud disclose that such referenced assembly comprises a manifest. But Evans discloses multiple representation of referenced assemblies by a structure with manifest depiction of assembly identification information, i.e. each referenced assembly in turn including manifest information as to represent itself (e.g. structure # - Fig. 11).

Nor does Renaud disclose that the assembly manifest includes a hash of the manifest of at least one of the referenced assembly. However, Evans discloses representing each referenced assembly with a manifest including a hash of the referenced assembly and Renaud discloses hash of each assembly manifest (*signature file 302* – Fig. 3b).

Based on the teachings by Evans to provide hash information on each referenced assembly referred to by the main assembly and by Renaud to make a hash for each assembly manifest from above, the motivation to provide the assembly manifest with referenced assemblies having each a manifest and to provide the assembly manifest with

a hash of such referenced assembly manifest would have been obvious for the same reasons as set forth in claim 10 above.

As per claim 28, the comparing of hash value limitation is rejected using the same rejection as set forth in claim 6 above because of the similarity in intent and purpose.

As per claim 29, Renaud does not teach a binding policy but discloses establishing and verification of signature with version information included and access permissions protocol (e.g. Fig. 5, 5a, 7) to enable trusted communication to bind usage of received files to the secure and trusted protocols; and also teaches determining which applet is to be trusted for execution (Fig. 6). Evans discloses binding assemblies to another version representation of another assembly (Evans: Fig. 11). It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify Renaud's method for integrity checking using trusted protocol of versioned assemblies such as to enhance it with the version dependency linking and correct version determining policy by Evans and effect it such to determine which version is to be executed when another version is resident on the executing environment because this would enable alleviating system errors from having uncontrolled versions dynamically competing in a same system.

As per claim 30, the limitation of related assembly is equivalent to referenced assembly as recited in claim 27 above, hence this claim is rejected with the same rejection as set forth in claim 27 above.

As per claim 31, Renaud teach about module (re claim 1) and Evans teaches about related assembly. The motivation to make such manifest of related assembly to

represent the module of Renaud would have been obvious in light of the rationale in claim 27 above.

As per claims 32-35, refer to claims 27, 28, 29, and 21 respectively.

6. Claims 14-17, and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051, and Evans et al., USPN: 5,805,899, as applied to claims 10 and 16 (for 14-17) and further in view of Buxton, USPN: 6,182,279.

As per claim 14, Renaud discloses hash of the identifiers of versioned modules in the assembly and suggests placing a signature (or hashed representation) of the manifest itself on top of the manifest with all other module identifiers at the end of the manifest (e.g. Fig. 3B), but **fails to disclose** providing a hash of the contents of such assembly at the end of the assembly. But this limitation to provide hash of the contents of modules has been addressed in claim 1 above, using Buxton's teaching. Hence it would have been obvious for one of ordinary skill in the art at the time the invention was made to add such hash of contents as taught by Buxton at the end of the assembly to enhance the lay out of the manifest in order to expedite information and facilitate efficiently the integrity checking as intended per module when resources at built time are available as mentioned in claim 1 above.

As per claim 15, the limitation as to determine whether the contents of the referenced assembly has been modified would also have been obvious in light of the rationale set forth in claim 6 which is further applied to the combination of Renaud/Evans of claim 10 because of the same benefits as set forth in claim 6.

As per claims 16 and 17, the rationales based on Buxton's teachings and used in the rejections of claims 7, 8, are herein respectively applied to the combination of Renaud/Evans for the same motivation as set forth therein correspondingly.

As per claim 22, Renaud discloses a readable medium (re claim 18) such medium comprising an assembly with a manifest containing a list of referenced assemblies. But such limitations have been addressed in claim 10 and 14 above; using the corresponding rationale as set forth therein to address the referenced assemblies limitation and hash of contents of referenced assemblies.

Response to Arguments

7. Applicant's arguments filed 4/15/2004 have been fully considered but they are not persuasive. Following are Examiner's remarks or replies in regard to all of the most representative points raised in Applicant's arguments.

As per rejection of claim 1-8, 18, and 23-25 under 35 USC 103(a):

(A) Applicants have submitted that the cited references do not teach 'ensuring integrity of assemblies necessary for proper operation ... mitigates errors that can occur ... a first application task modifies a module ... software damage' (Appl. Rmrks, pg. 8, bottom, pg. 9 top para). In response, it is noted that claim 1 only recites '... for facilitating integrity of an assembly employable by application programs during runtime'; and as interpreted, such claim cannot transpire into verifying integrity of components or modules or DLLs employed at runtime so that a modification to one module results in error without using hashing of the contents, as asserted by Applicants. The recited phrase '...assembly employable by application ...' at best can be interpreted as one of 2 possibilities: such assembly can be used for a runtime application and can also not be

used at all. The claim apparently amounts to a method for facilitating integrity of an assembly by providing an assembly with list of modules and a manifest. The claim's mere reciting of steps for providing an assembly and a manifest with hash of the contents would not qualify the invention as an novel method having concrete steps for effectively implementing the method of facilitating integrity, notably when the un-elaborated limitation 'facilitating integrity' amounts to but a broad concept. Thus, the claim body potentially amounts to a method not resulting in any concrete result in regard to the claimed method for integrity facilitating; and that could be a non-statutory subject matter. Applicants hence do not provide a specific description in the claim to support the above remarks, and yet assert that the references used do not fulfill the required elements that Applicants believe to be in the claims when in fact they are not. In light of the broadest reasonable interpretation, the rejection has pointed out a hash and a manifest with sections pointing to integrity checking for an assembly, and those are elements recited in the claim and fulfilled by Renaud in light of Buxton.

(B) Applicants have submitted that Renaud simply uses one-way hash functions on file names and such 'technique cannot support ... updated files ... DLL file have changed ... not the DLL file name' (Appl. Rmrks, 2nd para, pg. 9). The fact that Renaud does not provide a hash of contents has been addressed in the rejection. At the time the invention was made, one ordinary skill in the art would see it fit to provide the 'hash of the contents' teaching by Buxton to enhance the hash of file names by Renaud for specific advantages as specified in the rejection. Besides, the claims do not explicitly point out in what way the integrity checking using the hash of contents would have a clear novel advantage over (1) using hash of names or (2) over what has been suggested by the

combination set forth in the rejection. More importantly, the features concerning one-way hash or else for supporting verifying integrity of updated files upon which Applicants rely (i.e., one-way hash, updated files) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

(C) Applicants have submitted that Buxton 'simply provides a component system in which base applications ... distributed to another user ... same base applications' and that a 'container' is a stand alone application; and that 'there is no assembly containing a manifest with a list of modules ... Buxton deals with a stand alone application ... OLE controls ... There is no assembly with a manifest ... integrity checking' (Appl. Rmrks, pg. 10, 1st para). The rejection has pointed out that similar to a runtime gathering of modules as taught by Renaud, Buxton discloses templates as stage for storing loaded control components for a container application (e.g. *DLL storage 205, OLE Libraries, Template 208* - Fig. 2 – Note: Loading dynamic libraries and linking control objects to OLE/COM applications is equivalent to runtime linking); such templates, i.e. manifest, (*template* – Fig. 2) for associating components hashed contents for tracking of object identification (e.g. col. 12, lines 1-15; cols. 19-20); and integrity checking (e.g. Fig. 7-8B) using the hash of the contents of the components to be loaded, i.e. control components loaded by a *component loader* via a template execution with each component being a self-contained module (e.g. Fig. 2; Fig. 3A; col. 9, lines 8-25; col. 8, lines 49-63). Buxton is providing the hash of contents limitation for enhancing the hashing of file names by Renaud; and Buxton is having analogous teachings to Renaud's approach for

loading components for a runtime application. Since the claims do not explicitly describe in what way providing of hash of contents of modules distinguish over any other hash methods used in the references or prior art, the motivation for extending a hash of module name to a hash of content of modules as set forth in the rejection would have been found obvious for an ordinary skill in the art, when one is presented with 2 two teachings by Renaud and Buxton; and Applicants fail to point out why such combination results in failure or teach away from the references original intentions. The *container* by Buxton being a stand-alone application does not impact in the rationale used by the rejection; nor does the argument that Buxton deals with a stand-alone application and OLE controls have any effect in demising the rationale as set forth in the rejection. That is because the template and the components loader are to be equated to a manifest of components to be checked at runtime; and the hash of their contents are used for integrity checking, while the container is but a remote intended use at a higher level away from the loading of OLE control components by the template installer of Buxton's Figure 2. Since Renaud does not teach hash of contents, the rejection first establishes the analogous approach between Renaud and Buxton, then shows that Buxton has components to be loaded in some template (a manifest) construction interface (each components being viewed as a stand-alone module) and whose contents are hashed for integrity checking. The main purpose is to enhance the hash method by Renaud with the hash of contents by Buxton; and with such enhancement plus motivation as set forth in the rejection, a case of well-construed obviousness has been established. Applicants failed to point out why such combination results in failure or teach away from the references original intentions.

(D) Applicants have submitted that Examiner has exercised 20/20 hindsight based roadmap (Appl. Rmrks, pg. 10, bottom). The rejection has showed that Buxton by providing hash of contents has enhanced the hash method used by Renaud, and for one of ordinary skill in the art at the time the invention was made, it would be beneficial not only to hash a representation of a module but also the content of the module. It does not take much for one ordinary skill to come up with the reasoning that encoding a whole content of an object or file as via a hash would enforce a stricter integrity checking than just encoding its representation. The necessary steps of a prima facie case are listed in section C above. As pointed earlier, Applicants only assert that either Buxton or Renaud shows no teachings in light of what Applicants believed to be the inventive features while in fact Applicants fail to show how the proffered combination would be inappropriate, or in how exactly the invention as explicitly claimed is mapped with the foundation of the arguments being raised (e.g. one-way hash, updated data files, stand alone application etc.). As noted in section A, the important feature of the claim only amounts to steps for providing an assembly and a manifest for listing modules with hash of the contents, a method whose purpose for integrity facilitating remains open to broad interpretation and whose feature called 'assembly' can or can not be employed at runtime of an application program. Such feature has been addressed in the corresponding rejection.

As per rejection of claim 10-13, and 27-35 under 35 USC 103(a):

(E) Applicants have submitted that for claims 10, 27, and 30, the cited references do not teach 'ensuring integrity of assemblies necessary for proper operation ... a hash to ascertain whether contents of modules ... have been modified' (Appl. Rmrks, pg. 11, 4th para). It is a known concept in the art that using hash or checksum/CRC is for

performing verification whether a content would not be the same with respect to the original content prior to a transaction or a application usage. Besides, the exact wording in the above claims does not remotely include the elements thus listed by Applicants in the arguments. For example, claim 10 only recites a "method for facilitating integrity checking ... providing an assembly with a manifest ... the manifest with a hash of a manifest ... referenced assembly ...". There is no "...ascertaining whether contents ... been modified' as asserted by Applicants. What is lacking in the combination Renaud/Buxton has been addressed in the rationale using Evans.

(F) Applicants have submitted that Evans discloses 'creating a dynamic executable ... dynamic executable ... is not equivalent to an assembly ... cannot be equivalent to a list of referenced assemblies ... the hash is not a hash of contents of a manifest ... claimed invention' (Appl. Rmrks, pg. 11, bottom, pg. 12 1st para). What is at issue is that Renaud does not disclose that the manifest contains a list of referenced assemblies that the assembly depends on; nor does Renaud disclose a manifest with a hash of a manifest of one referenced assembly of the list of referenced assemblies. Thus, in a method to provide version checking of object files in a linking process within the creation of object executable, Evans discloses dependencies between modules represented by special file format, or assemblies, and such assemblies are referenced for such linking and version checking by other assemblies. The rationale as set forth in the rejection addresses the dependency among assemblies being loaded or gathered for a runtime application; and since this has been suggested by Renaud, the motivation would have been obvious in view of teachings by Evans for using some assembly structures to point to other assemblies in providing a manifest with the list of referenced assemblies (or files or

objects) and using therein pointer information to point to the hash representation of the manifest of the assembly referred to by such pointer. Both Evans and Renaud gather assemblies and manifest thereof that list modules whose integrity are to be checked with some hash techniques, and the fact that both are for a runtime module checking reads on the recited limitation such as 'assemblies employable by application programs during runtime'. The pointing mechanism used by Evans do read on list of referenced assemblies, because each special format file by Evans (ELF file/assembly) encompasses a list of sections which again point to other assemblies as pointed out in Fig. 16 by Evans. There is no 'dynamic executable' involved in the rationale as to use of one assembly to point to another assembly, and this feature by Evans is what has been used in the rejection in order to complement what is missing in Renaud's approach of linking modules. Further, there is no reciting in the claim of hash of contents of a manifest as raised by Applicants. The hash of contents has been addressed earlier in claim 18. Applicants is working to show that one reference does not fulfill what Applicants believe to his the invention features when in fact those features are either absent in the claim or already have been addressed by an earlier claim rejection; thus Applicants miss the main focus of the rationale being constructed to address what is missing by Renaud when Evans' teachings can be utilized on the standpoint of one ordinary skill in the art when presented with both teachings.

As per rejection of claim 14-17, and 22 under 35 USC 103(a):

(G) Applicants have submitted that Buxton deals with stand-alone application and OLE controls and does not teach a hash of the contents of modules; and further submitted that Evans discloses a dynamic executable for shared object, none of it being equivalent

to a list of referenced assemblies (Appl. Rmrks, pg. 12, bottom, pg. 13, top 2 paras). Claim 14 requires a hash of contents of the assembly at the end of an assembly. In view that Renaud fails to provide a hash of the contents of such assembly at the end of the assembly, the rejection has taken the approach of using Buxton to provide hash of the contents of modules as has been addressed in claim 1 above. The same Applicants' arguments as to how Buxton's stand-alone OLE container fails to provide an assembly during a runtime application and how Evans fails to teach list of referenced assemblies at least have been addressed in sections C and F above; and further do not amount to pointing out how the rationale as set forth in claim 14 would be inappropriate in terms of the combined features and the motivation to combine.

Applicants have submitted that Examiner had used 20/20 hindsight (Appl. Rmrks, pg. 13, 3rd para). In response to such argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971). As pointed in the rejection, Examiner has relied solely on the references used and did not use the teachings from the invention by default of prior art teachings at hand. Besides, Applicants do not provide specifics as to how such hindsight has been employed in the course of Examiner's rejection, hence this argument amounts to mere assertions without convincing support as to why Examiner's rationale as set forth would be inappropriate.

As per rejection of claims 9, 19-21, and 26 under 35 USC 103(a):

(H) Applicants have submitted that Buxton does not teach or suggest an assembly containing a manifest with a list of modules (Appl. Rmrks, pg. 14, 2nd para). Examiner's interpretation of Buxton version of list of modules is that a template is for showing how different components have been fetched into a building interface in order to achieve a linking and verification scheme, and as such a template is a form listing components that have loaded for further applications. Besides, in claim 9, the issue evolves around providing a hash of a manifest. And as set forth in the rejection, the combined teachings by Renaud, Buxton, and Evans has been put into a rationale so as to render this feature obvious using the motivation as put forth therein.

(I) In the 3rd para. Appl. Rmrks, pg. 14, the limitation raised by Applicants for claim 21 for addressing the hash of contents has been addressed earlier (see section C).

As to the remark concerning the **Official Notice**, Examiner likes to exhibit the following:

Buxton uses a DLL for storing the control components for a COM application (Fig. 2)

USPN: 5,802,368 (e.g. Fig. 3) teaches DLL for task switching in Windows applications.

USPN: 6,445,973 (e.g. Fig. 2-4) teaches using library DLL for screen switching.

USPN: 6,212,673 (e.g. Fig. 6) teaches a DLL for component builder.

In conclusion, for the reasons mentioned above, the claims will stand rejected as set in the current rejection.

Conclusion

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please consult Examiner before using this number)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
July 2, 2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100